08-18-00 A

SKJERVEN MORRILL MACPHERSON LLP

Docket No.: SP-2616 US

August 16, 2000

jc867 U.S. PTO 08/16/00

JC897 U.S. PTO 09/640901 08/16/00

Box Patent Application
Commissioner For Patents
Washington, D. C. 20231

Enclosed herewith for filing is a patent application, as follows:

Inventor(s): Subramania Sudharsanan, Jeff Meng Wah Chan, Michael F. Deering, Marc Tremblay, Scott Nelson

Title: Accelerated Graphic Operations

X Return Receipt Postcard
X This Transmittal Letter (in duplicate)
3 page(s) Declaration For Patent Application and Power of Attorney (unsigned)
17 page(s) Specification (not including claims)
7 page(s) Claims
1 page Abstract
10 Sheet(s) of Drawings

CLAIMS AS FILED

For	Number Filed			Number Extra		Rate			Basic Fee
								$	$690.00
Total Claims	26	-20	=	6	x	$18.00	=	$	108.00
Independent Claims	10	-3	=	7	x	$78.00	=	$	546.00

☐ Fee of ____ for the first filing of one or more multiple dependent claims per application $

☐ Fee for Request for Extension of Time $

Please make the following charges to Deposit Account 19-2386:

☒ Total fee for filing the patent application in the amount of $ 1,344.00

☒ The Commissioner is hereby authorized to charge any additional fees which may be required, or credit any overpayment to Deposit Account 19-2386.

EXPRESS MAIL LABEL NO:
EM 263 038 970 US

Respectfully submitted,

Fabio E. Marino
Attorney for Applicant(s)
Reg. No. 43,339

25 Metro Drive • Suite 700 • San Jose, CA 95110 • Phone (408) 453-9200 • Fax (408) 453-7979
San Francisco, CA • Austin, TX • Newport Beach, CA

EXPRESS MAIL LABEL NO:
EM 263 038 970 US

ACCELERATED GRAPHIC OPERATIONS

Subramania Sudharsanan
Jeffrey Meng Wah Chan
Michael F. Deering
Marc Tremblay
Scott Nelson

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to processors and, more particularly to instructions for use with processors.

Related Art

In order to support speech and audio processing, signal processing and 2-D and 3-D graphics, processors must be able to support fast graphics operations. However, prior art general purpose processors have provided little or no hardware support for this type of operations. By contrast, special purpose graphics and media processors provide hardware support for specialized operations. As a result, using prior art processors, graphical operations were performed mostly with the aid of a specialized graphics/media processor.

As the demand for graphics/media support in general purpose processors rises, hardware acceleration of these operations becomes more and more important.

As a result, there is a need for a general purpose processor that allows for efficient processing of these operations.

SUMMARY OF THE INVENTION

The present invention provides a method and apparatus for efficiently performing graphic operations. This is accomplished by providing a

processor that supports any combination of the following instructions: parallel multiply-add, conditional pick, parallel averaging, parallel power, parallel reciprocal square root and parallel shifts. In some embodiments, the results of these operations are further saturated within specified numerical ranges.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1A is a schematic block diagram illustrating a single integrated circuit chip implementation of a processor in accordance with an embodiment of the present invention.

Fig. 1B is a schematic block diagram showing the core of the processor.

Fig. 2A is a block diagram of a register file of the processor of Fig. 1B.

Fig. 2B is a block diagram of a register of the register file of Fig. 2A.

Fig. 3A is a block diagram showing instruction formats for four-operand instructions supported by the processor of Fig. 1B.

Fig. 3B is a block diagram showing instruction formats for three-operand instructions supported by the processor of Fig. 1B.

Fig. 4A is a block diagram showing an instruction format for a parallel multiply-add instruction supported by the processor of Fig. 1B.

Fig. 4B is a block diagram showing an instruction format for a conditional pick instruction supported by the processor of Fig. 1B.

Fig. 4C is a block diagram showing an instruction format for a parallel mean instruction supported by the processor of Fig. 1B.

Fig. 4D is a block diagram showing instruction formats for a parallel logical shift left instruction supported by the processor of Fig. 1B.

Fig. 4E is a block diagram showing instruction formats for a parallel arithmetic shift right instruction supported by the processor of Fig. 1B.

Fig. 4F is a block diagram showing instruction formats for a parallel logical shift right instruction supported by the processor of Fig. 1B.

Fig. 5 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing the pmuladd instruction of Fig. 4A.

Fig. 6 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing the cpickz instruction of Fig. 4B.

Fig. 7 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing the pmean instruction of Fig. 4C.

Fig. 8A is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing any of the parallel shift instructions of Figs. 4D, 4E or 4F, when operands are register references.

Fig. 8B is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing any of the parallel shift instructions of Figs. 4D, 4E or 4F, when one of the operands is an immediate value.

DETAILED DESCRIPTION OF THE INVENTION

A processor in accordance to the principles of the present invention is illustrated in Fig. 1.

Referring to Fig. 1A, a schematic block diagram illustrates a single integrated circuit chip

implementation of a processor 100 that includes a memory interface 102, a geometry decompressor 104, two media processing units 110 and 112, a shared data cache 106, and several interface controllers. The interface controllers support an interactive graphics environment with real-time constraints by integrating fundamental components of memory, graphics, and input/output bridge functionality on a single die. The components are mutually linked and closely linked to the processor core with high bandwidth, low-latency communication channels to manage multiple high-bandwidth data streams efficiently and with a low response time. The interface controllers include a an UltraPort Architecture Interconnect (UPA) controller 116 and a peripheral component interconnect (PCI) controller 120. The illustrative memory interface 102 is a direct Rambus dynamic RAM (DRDRAM) controller. The shared data cache 106 is a dual-ported storage that is shared among the media processing units 110 and 112 with one port allocated to each media processing unit. The data cache 106 is four-way set associative, follows a write-back protocol, and supports hits in the fill buffer (not shown). The data cache 106 allows fast data sharing and eliminates the need for a complex, error-prone cache coherency protocol between the media processing units 110 and 112.

Two media processing units 110 and 112 are included in a single integrated circuit chip to support an execution environment exploiting thread level parallelism in which two independent threads can execute simultaneously. The threads may arise from any sources such as the same application, different applications, the operating system, or the runtime environment. Parallelism is exploited at the thread

level since parallelism is rare beyond four, or even two, instructions per cycle in general purpose code. For example, the illustrative processor 100 is an eight-wide machine with eight execution units for executing instructions. Typical "general-purpose" processing code has an instruction level parallelism of about two so that, on average, most (about six) of the eight execution units would be idle at any time. The illustrative processor 100 employs thread level parallelism and operates on two independent threads, possibly attaining twice the performance of a processor having the same resources and clock rate but utilizing traditional non-thread parallelism.

Although the processor 100 shown in Fig. 1A includes two processing units on an integrated circuit chip, the architecture is highly scaleable so that one to several closely-coupled processors may be formed in a message-based coherent architecture and resident on the same die to process multiple threads of execution. Thus, in the processor 100, a limitation on the number of processors formed on a single die thus arises from capacity constraints of integrated circuit technology rather than from architectural constraints relating to the interactions and interconnections between processors.

Referring to Fig. 1B, a schematic block diagram shows the core of the processor 100. The media processing units 110 and 112 each include an instruction cache 210, an instruction aligner 212, an instruction buffer 214, a pipeline control unit 226, a split register file 216, a plurality of execution units, and a load/store unit 218. In the illustrative processor 100, the media processing units 110 and 112 use a plurality of execution units for executing

instructions. The execution units for a media processing unit 110 include three media functional units (MFU) 222 and one general functional unit (GFU) 220. The media functional units 222 are multiple single-instruction-multiple-data (MSIMD) functional units. Each of the media functional units 222 is capable of processing parallel 16-bit components. Various parallel 16-bit operations supply the single-instruction-multiple-data capability for the processor 100 including add, multiply-add, shift, compare, and the like. The media functional units 222 operate in combination as tightly-coupled digital signal processors (DSPs). Each media functional unit 222 has a separate and individual sub-instruction stream, but all three media functional units 222 execute synchronously so that the subinstructions progress lock-step through pipeline stages.

The general functional unit 220 is a RISC processor capable of executing arithmetic logic unit (ALU) operations, loads and stores, branches, and various specialized and esoteric functions such as parallel power operations, reciprocal squareroot operations, and many others. The general functional unit 220 supports less common parallel operations such as the parallel reciprocal square root instruction.

The pipeline control unit 226 is connected between the instruction buffer 214 and the functional units and schedules the transfer of instructions to the functional units. The pipeline control unit 226 also receives status signals from the functional units and the load/store unit 218 and uses the status signals to perform several control functions. The pipeline control unit 226 maintains a scoreboard, generates stalls and bypass controls. The pipeline control unit

226 also generates traps and maintains special registers.

Each media processing unit 110 and 112 includes a split register file 216, a single logical register file including 224 32-bit registers. The split register file 216 is split into a plurality of register file segments 224 to form a multi-ported structure that is replicated to reduce the integrated circuit die area and to reduce access time. A separate register file segment 224 is allocated to each of the media functional units 222 and the general functional unit 220. In the illustrative embodiment, each register file segment 224 has 128 32-bit registers. The first 96 registers (0-95) in the register file segment 224 are global registers. All functional units can write to the 96 global registers. The global registers are coherent across all functional units (MFU and GFU) so that any write operation to a global register by any functional unit is broadcast to all register file segments 224. Registers 96-127 in the register file segments 224 are local registers. Local registers allocated to a functional unit are not accessible or "visible" to other functional units.

The media processing units 110 and 112 are highly structured computation blocks that execute software-scheduled data computation operations with fixed, deterministic and relatively short instruction latencies, operational characteristics yielding simplification in both function and cycle time. The operational characteristics support multiple instruction issue through a pragmatic very large instruction word (VLIW) approach that avoids hardware interlocks to account for software that does not schedule operations properly. Such hardware interlocks

are typically complex, error-prone, and create multiple critical paths. A VLIW instruction word always includes one instruction that executes in the general functional unit (GFU) 220 and from zero to three instructions that execute in the media functional units (MFU) 222. A MFU instruction field within the VLIW instruction word includes an operation code (opcode) field, three source register (or immediate) fields, and one destination register field.

Instructions are executed in-order in the processor 100 but loads can finish out-of-order with respect to other instructions and with respect to other loads, allowing loads to be moved up in the instruction stream so that data can be streamed from main memory. The execution model eliminates the usage and overhead resources of an instruction window, reservation stations, a re-order buffer, or other blocks for handling instruction ordering. Elimination of the instruction ordering structures and overhead resources is highly advantageous since the eliminated blocks typically consume a large portion of an integrated circuit die. For example, the eliminated blocks consume about 30% of the die area of a Pentium II processor.

Processor 100 is further described in co-pending application Ser. No. 09/204,480, entitled "A Multiple-Thread Processor for Threaded Software Applications" by Marc Tremblay and William Joy, filed on Dec. 3, 1998, which is herein incorporated by reference in its entirety.

The structure of a register file of the processor of Fig. 1B is illustrated in Fig. 2A. The register file is made up of an arbitrary number of registers R0, R1, R2 . . . Rn. Each of registers R0, R1, R2 . . .

Rn, in turn has an arbitrary number of bits n, as shown in Fig. 2B. In one embodiment, the number of bits in each of registers R0, R1, R2 . . . Rn is 32. However, those skilled in the art realize that the principles of the present invention can be applied to an arbitrary number of registers each having an arbitrary number of bits. Accordingly, the present invention is not limited to any particular number of registers or bits per register.

Fig. 3A illustrates an instruction format for four-operand instructions supported by the processor of Fig. 1B. The instruction format has a 4-bit opcode and four 7-bit operands. The first of the operands is a reference to a destination register (RD) for the instruction. The second operand, in turn, is a reference to a first source register for the instruction (RS1). The third operand is a reference to a second source register for the instruction (RS2) and the fourth operand is a reference to a third source register for the instruction(RS3).

Fig. 3B illustrates two instruction formats for three-operand instructions supported by the processor of Fig. 1B. Each instruction format has an 11-bit opcode and three 7-bit operands. The first of the operands is a reference to a destination register (RD) for the instruction. The second operand, in turn, is a reference to a first source register for the instruction (RS1). Finally, the third operand can be a references to a second (RS2) source register or an immediate value to be used in the instruction.

Fig. 4A illustrates an instruction format for a parallel multiply-add instruction (pmuladd) supported by the processor of Fig. 1B, in accordance to the present invention. The pmuladd instruction uses the

four-operand instruction format of Fig. 3A, namely a format in which no immediate values are used. Rather, all operands are references to registers in the register file of the processor. Fig. 4B illustrates an instruction format for a conditional pick instruction (cpickz) supported by the processor of Fig. 1B. The cpickz instruction uses the four-operand instruction format of Fig. 3A. Fig. 4C illustrates an instruction format for a parallel mean instruction (pmean) supported by the processor of Fig. 1B. The pmean instruction uses the first of the three-operand instruction formats of Fig. 3B, namely a format in which no immediate values are used. Fig. 4D illustrates instruction formats for a pshll instruction supported by the processor of Fig. 1B. The pshll instruction uses either of the three-operand instruction formats of Fig. 3B. Fig. 4E illustrates instruction formats for a pshra instruction supported by the processor of Fig. 1B. The pshra instruction uses either of the three-operand instruction formats of Fig. 3B. Fig. 4F illustrates instruction formats for a pshrl instruction supported by the processor of Fig. 1B. The pshrl instruction uses either of the three-operand instruction formats of Fig. 3B.

Fig. 5 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing a parallel multiply-add operation. The pmuladd instruction treats values stored in the source registers as each having two 16-bit fixed-point components. For example, in Fig. 5, bits 0..15 of the values stored in registers RS1, RS2 and RS3 comprise the first fixed-point operands and bits 16..31 comprise the second fixed-point operands. The multiply-add operation is then carried out separately on the first

operands and on the second operands. As a result, after the execution of a pmuladd instruction, the value stored in register RD represents two 16 bit fixed-point values, one representing a value calculated by multiplying the first fixed-point operand of RS1 by the first fixed-point operand of RS2 and adding the first fixed-point operand of RS3, and the other representing a value calculated by multiplying the second fixed-point operand of RS1 by the second fixed-point operand of RS2 and adding the second fixed-point operand of RS3.

In the implementation shown in Fig. 5, when executing a pmuladd instruction, the processor routes the value of bits 0..15 (high-order bits) of RS1 and RS2 to respective input ports of multiplier 510, while the value of bits 16..31 (low-order bits) of RS1 and RS2 are routed to respective input ports of multiplier 520. After a time delay for propagating the input values through multipliers 510 and 520, values on respective output ports of multipliers 510 and 520 are routed to respective input ports of adders 530 and 540. The value of bits 0..15 of RS3 is then routed to the other input port of adder 530 and the values of bits 16..31 of RS3 are routed to the other input port of adder 540. After a time delay for propagating the input values through adders 530 and 540, a value on an output port of adder 530 is stored in bits 0..15 of register RD, while a value on an output port of adder 540 is stored in bits 16..31 of register RD.

The results depend on the values of two mode/format bits. The operands can be either in fixed-point format or in integer format. As shown in Table 1, when the mode bits have 00 and 01 values, both the operands and the result are treated as two's complement

16-bit integer values. When the mode bits have a 10 value, the operands and the result are treated as S.15 fixed-point values. Finally, if the mode bits have a 11 value, the operands and the result are treated as S2.13 fixed point values. Hence, depending on the value of the mode bits the appropriate bits from the multiplier results are supplied to the adder.

Moreover, the processor of Fig. 1B supports saturation functions to be performed during pmuladd, padd and psub operations. Four different saturation modes are provided, as shown in Table 1 below.

mode	format	Bounds	
		Low	High
00	Integer	0000000000000000	0111111111111111
01	Integer	1000000000000000	0111111111111111
10	S.15	1000000000000000	0111111111111111
11	S2.13	1110000000000000	0010000000000000

Table 1.

Saturation modes 00 and 01 in Table 1 represent two's complement 16-bit integers. Mode 10 represents an S.15 fixed point notation, while mode 11 represents an S2.13 fixed point format. In both of these notations, the S bit is part of the integer part of the fixed point number. For example, an S2.13 number has a 3-bit integer part and a 13-bit fractional part.

Using mode 00, the parallel muladd with saturation

instruction will produce a value between 0 and 2^{15}-1, inclusive. If the results exceed these bounds, they are "capped" at the upper bound. Similarly, mode 01 limits the result to between -2^{15} and 2^{15}-1, inclusive. Modes 10 and 11 represent saturation for fixed point formats. Table 1 summarizes the limits or bounds for all four modes.

Execution of these instructions is pipelined to achieve a throughput of one instruction per cycle.

Fig. 6 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing a conditional pick operation. The cpickz instruction compares a value stored in register RS1 to a zero value and depending on the outcome of the comparison copies the values stored in either register RS2 or register RS3 into register RD.

In the implementation of Fig. 6, when executing a cpickz instruction, the processor routes a value stored in register RS1 to an input port of comparator 610. A zero value is supplied on the other input port of comparator 610. After a time delay for propagating the input values through comparator 610, a value on an output port of comparator 610 is routed to a control port of multiplexer 620. Meanwhile, values stored in registers RS2 and RS3 are routed by the processor to respective input ports of multiplexer 620. After a time delay for propagating input values through multiplexer 620, a value on an output port of multiplexer 620 is stored in register RD.

As a result, after the execution of a cpickz instruction, the value stored in register RD is a copy of the value stored in register RS2 if the value stored in register RS1 is not equal to 0. Alternatively, if the value stored in register RS1 is equal to 0, the

value stored in register RD is a copy of the value stored in register RS3.

Fig. 7 is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing a parallel averaging operation. The pmean instruction treats values stored in the source registers as each having two 16-bit integer components. For example, in Fig. 7, bits 0..15 of the values stored in registers RS1 and RS2 comprise the first integer operands and bits 16..31 comprise the second integer operands. The averaging operation is then carried out separately on the first operands and on the second operands. As a result, after the execution of a pmean instruction, the value stored in register RD represents two 16 bit integer values, one representing a value calculated by averaging the first integer operand of RS1 with the first integer operand of RS2, and the other representing a value calculated by averaging the second integer operand of RS1 with the second integer operand of RS2.

In the implementation of Fig. 7, when executing a pmean instruction, the processor routes values stored in bits 0..15 of registers RS1 and RS2 to respective input ports of adder 710. Meanwhile, values stored in bits 16..31 of registers RS1 and RS2 are routed to respective input ports of adder 720. After a time delay for propagating the input values through adders 710 and 720, values on respective output ports of adders 710 and 720 are routed to respective input ports of adders 730 and 740. A 1 value is supplied on respective input ports of adders 730 and 740. After a time delay for propagating the input values through adders 730 and 740, output values on respective ports of adders 730 and 740 are routed to respective input

ports of right shifters 750 and 760. A logical one value is supplied on respective control ports of right shifters 750 and 760. After a time delay for propagating the input values through right shifters 750 and 760, a value on an output port of right shifter 750 is copied into bits 0..15 of register RD and a value on an output port of right shifter 760 is copied into bits 16..31 of register RD.

Fig. 8A is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing a parallel shift operation, when all operands are provided as register references. The pshll instruction treats values stored in the source registers as each having two 16-bit integer components. For example, in Fig. 8A, bits 0..15 of the values stored in registers RS1 and RS2 comprise the first integer operands and bits 16..31 comprise the second integer operands. The logical shift left operation is then carried out separately on the first operands and on the second operands. As a result, after the execution of a pshll instruction, the value stored in register RD represents two 16 bit integer values, one representing a value calculated by performing a logical shift left of the first integer operand of RS1 by a number of bits specified by the first integer operand of RS2, and the other representing a value calculated by performing a logical shift left on the second integer operand of RS1 by a number of bits specified by the second integer operand of RS2.

In the implementation of Fig. 8A, when executing the pshll instruction, the processor routes the value stored in bits 0..15 of register RS1 to an input port of shifter 810. Meanwhile, the value stored in bits 16..31 of register RS1 are routed to an input port of

shifter 820. The processor also routes bits 0..3 of registers RS1 and RS2 to respective select ports of shifters 810 and 820. After a time delay for propagating the input values through shifters 810 and 820, a value on an output port of shifter 810 is copied into bits 0..15 of register RD and a value on an output port of shifter 820 is copied into bits 16..31 of register RD.

Fig. 8B is a block diagram of one implementation of the circuitry within MFUs 222 of the processor of Fig. 1B for performing a parallel shift operation, when the second source operand is provided as an immediate value. The functioning of the circuitry of Fig. 8B is identical to that of the circuitry of Fig. 8A, except that bits 0..3 of the immediate value are routed to both input ports of shifters 810 and 820.

The operation of the circuitry of Figs. 8A and 8B during execution of a pshra or a pshrl instructions is identical to the one described for the execution of a pshll instruction, except that shifters 810 and 820 perform an arithmetic shift right or a logical shift right operations, respectively.

In addition, the processor of Fig. 1B supports a parallel power instruction, ppower and a parallel reciprocal square root instruction precsqrt. The ppower and precsqrt instruction treat the values stored in the source registers as a pair of fixed-point (rather than integer) components. Therefore, the value stored in bits 0..15 of register RD after the execution of a ppower instruction represent a value calculated by raising the value stored in bits 0..15 of register RS1 to a power specified by the value stored in bits 0..15 of register RS2. Similarly, the value stored in bits 16..31 of register RD after the execution of a ppower

instruction represent a value calculated by raising the value stored in bits 16..31 of register RS1 to a power specified by the value stored in bits 16..31 of register RS2. The pair of values stored in register RD after the execution of a precsqrt instruction are calculated using a similar process to the one described for the ppower instruction, except that the reciprocal square roots of the pairs of values stored in register RS1 are computed, rather than a power.

The precsqrt instruction is further described in co-pending application Ser. No. 09/240,977 titled "Parallel Fixed Point Square Root And Reciprocal Square Root Computation Unit In A Processor" by Ravi Shankar and Subramania Sudharsanan, which is incorporated by reference herein in its entirety.

Embodiments described above illustrate but do not limit the invention. In particular, the invention is not limited by any number of registers or immediate values specified by the instructions. In addition, the invention is not limited to any particular hardware implementation. Those skilled in the art realize that alternative hardware implementation can be employed in lieu of the one described herein in accordance to the principles of the present invention. Other embodiments and variations are within the scope of the invention, as defined by the following claims.

CLAIMS

1. A method of executing a single instruction parallel multiply-add function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel multiply-add instruction;

providing the processor with a first, a second and a third value, wherein each of the values comprises two or more operand components;

multiplying first operand components of the first and the second values to generate a first intermediate value;

multiplying second operand components of the first and the second values to generate a second intermediate value;

adding a first operand component of the third value to the first intermediate value to generate a first result value;

adding a second operand component of the third value to the second intermediate value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

2. The method of claim 1, wherein the first, second and third values are stored in respective source registers of the processor specified by the parallel multiply-add instruction, and the first and the second result values are stored in a destination register of the processor specified by the parallel multiply-add instruction.

3. The method of claim 2, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

4. The method of claim 1, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction every 2 cycles.

5. A method of executing a single instruction conditional pick function on a processor, the method comprising:

providing the processor with an opcode indicating a conditional pick instruction;

providing the processor with a first, a second and a third value;

comparing the first value to a reference value;

determining, based upon the comparing, whether the first value is equal to the reference value;

storing the second value in a result location if the first value is equal to the reference value; and

storing the third value in a result location if the first value is not equal to the reference value.

6. The method of claim 5, wherein the first, second and third values are stored in respective source registers of the processor specified by the conditional pick instruction, and the second and the third values are stored in a destination register of the processor specified by the conditional pick instruction.

7. The method of claim 5, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

8. A method of executing a single instruction parallel averaging function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel averaging instruction;

providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;

adding first operand components of the first and the second values to generate a first intermediate value;

adding second operand components of the first and the second values to generate a second intermediate value;

incrementing the first intermediate value by one to generate a third intermediate value;

incrementing the second intermediate value by one to generate a fourth intermediate value;

shifting the third intermediate value to generate a first result value;

shifting the fourth intermediate value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

9. The method of claim 8, wherein the first and the second values are stored in respective source

registers of the processor specified by the parallel averaging instruction.

10. The method of claim 8, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel averaging instruction.

11. The method of claim 10, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

12. The method of claim 8, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

13. A method of executing a single instruction parallel shift function on a processor, the method comprising:

providing the processor with an opcode indicating a parallel shift instruction;

providing the processor with a first and a second value, wherein each of the values comprises two or more operand components;

shifting the first operand component of the first value by a number of bits equal to a value of the first operand component of the second value to generate a first result value;

shifting the second operand component of the first value by a number of bits equal to a value of the second operand component of the second value to generate a second result value;

storing the first result value in a first portion of a result location; and

storing the second result value in a second portion of the result location.

14. The method of claim 13, wherein the first and the second values are stored in respective source registers of the processor specified by the parallel shift instruction.

15. The method of claim 13, wherein the first and the second result values are stored in a destination register of the processor specified by the parallel shift instruction.

16. The method of claim 15, the first result value is stored in the high-order bits of the destination register and the second result value is stored in the low-order bits of the destination register.

17. The method of claim 13, wherein the processor is pipelined and the single instruction is executed with a throughput of one instruction per cycle.

18. A general purpose processor comprising:

a file register;

an instruction fetch unit; and

decoding circuitry;

wherein the processor supports a parallel multiply-add instruction.

19. The general purpose processor of claim 18, wherein the parallel multiply-add instruction operate

on either integer or fixed point operands.

20. The general purpose processor of claim 19, wherein the results of the parallel multiply-add instruction are saturated.

21. The general purpose processor of claim 19, wherein the parallel multiply-add instruction further provides multiple saturation modes.

22. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a conditional pick instruction.

23. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a parallel averaging instruction.

24. A general purpose processor comprising:
a file register;
an instruction fetch unit; and
decoding circuitry;
wherein the processor supports a parallel shift instruction.

25. A general purpose processor comprising:
a file register;
an instruction fetch unit; and

decoding circuitry;

wherein the processor supports a parallel power instruction.

26. A general purpose processor comprising:

a file register;

an instruction fetch unit; and

decoding circuitry;

wherein the processor supports a parallel reciprocal square root instruction.

ACCELERATED GRAPHIC OPERATIONS

Subramania Sudharsanan
Jeffrey Meng Wah Chan
Michael F. Deering
Marc Tremblay
Scott Nelson

ABSTRACT OF THE INVENTION

A method and apparatus for efficiently performing graphic operations are provided. This is accomplished by providing a processor that supports any combination of the following instructions: parallel multiply-add, conditional pick, parallel averaging, parallel power, parallel reciprocal square root and parallel shifts.

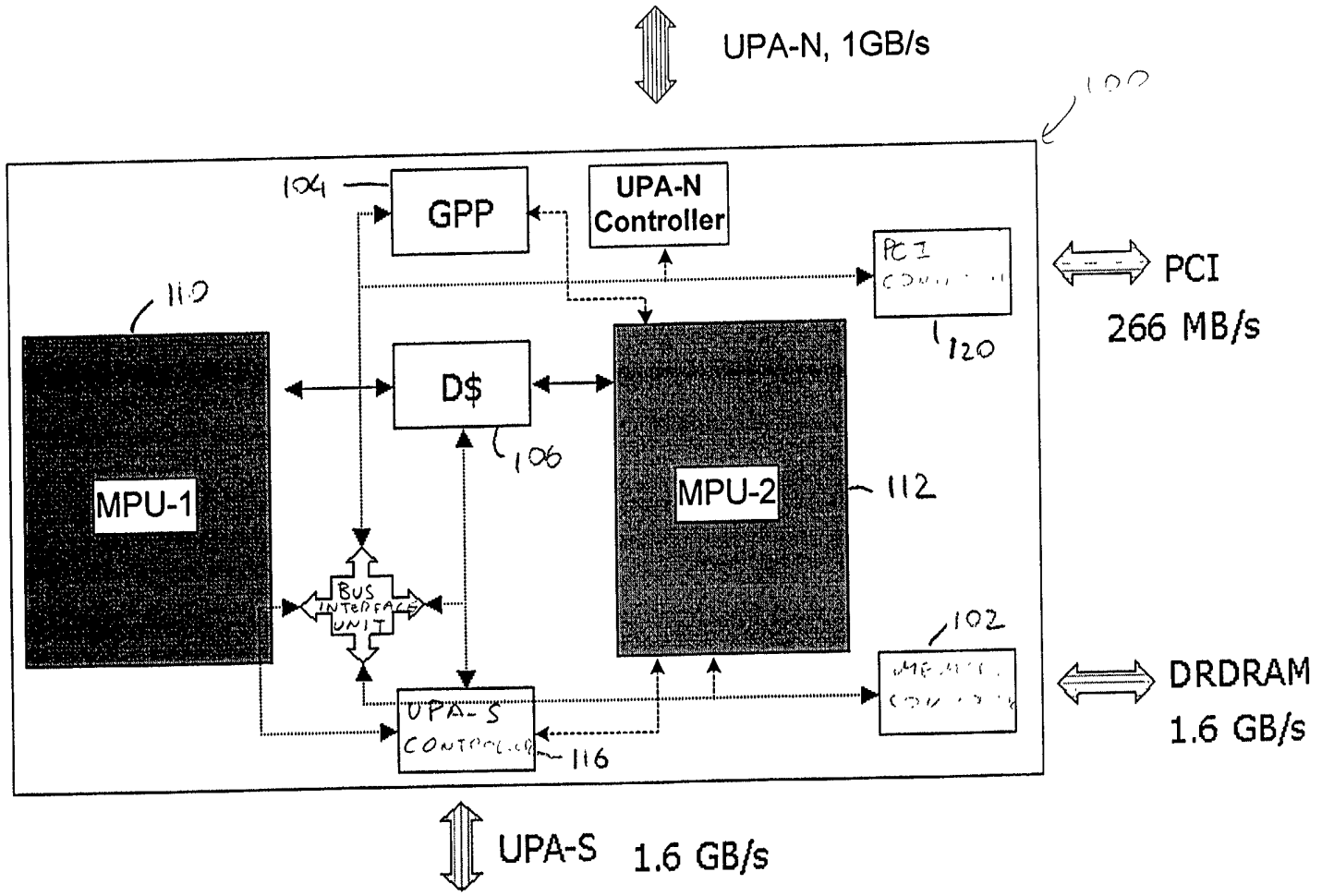
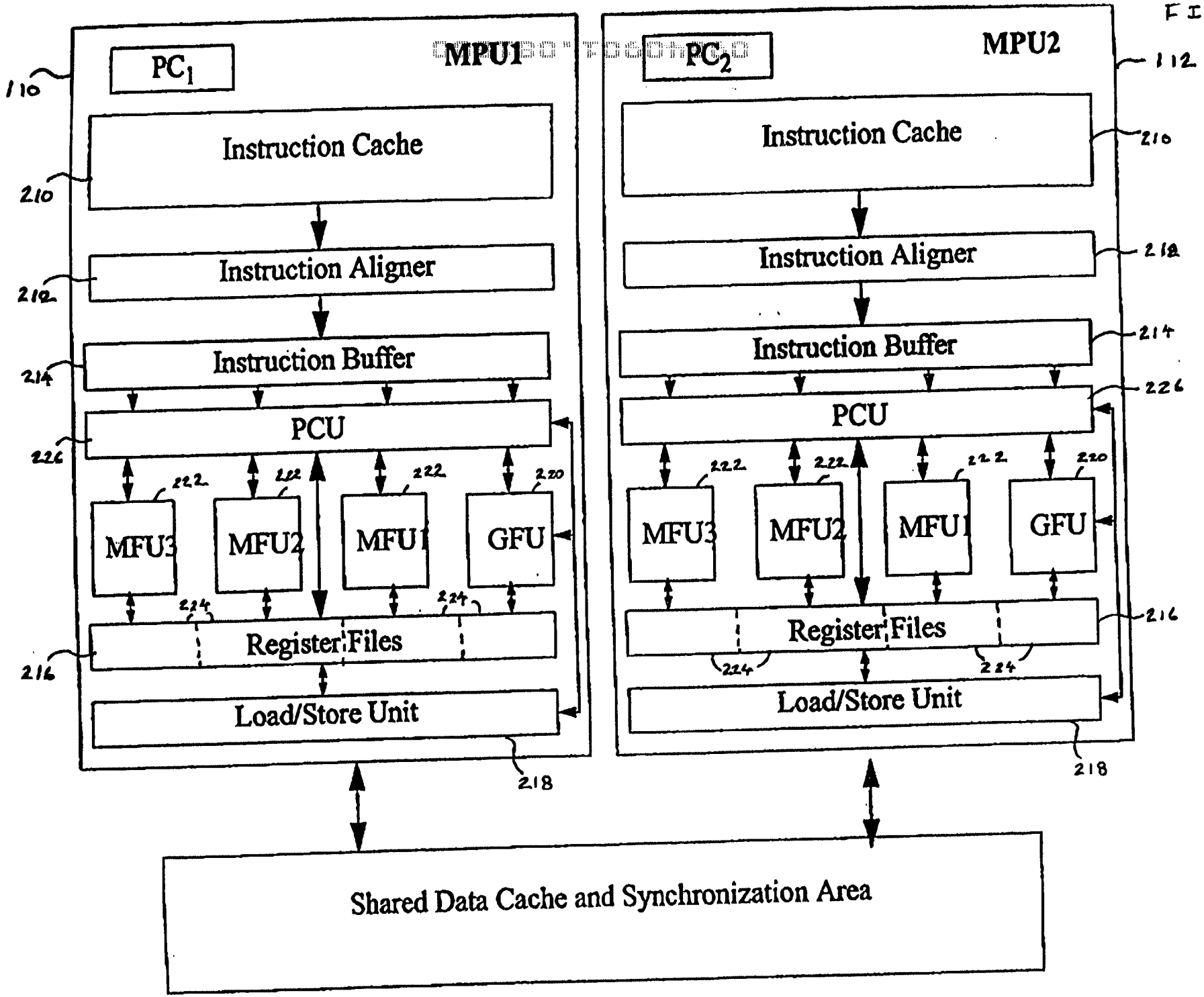

UPA-N, 1GB/s
100
104
GPP
UPA-N Controller
PCI
120
PCI
266 MB/s
110
D$
106
MPU-1
MPU-2
112
BUS INTERFACE UNIT
102
UPA-S CONTROLLER
116
DRDRAM
1.6 GB/s
UPA-S 1.6 GB/s

Fig. 1A

FIG. 1B


MPU1
PC1
110
Instruction Cache
210
Instruction Aligner
212
Instruction Buffer
214
PCU
226
MFU3
MFU2
MFU1
GFU
222
220
224
Register Files
216
Load/Store Unit
218
MPU2
PC2
112
Instruction Cache
210
Instruction Aligner
212
Instruction Buffer
214
PCU
226
MFU3
MFU2
MFU1
GFU
222
220
224
Register Files
216
Load/Store Unit
218
Shared Data Cache and Synchronization Area

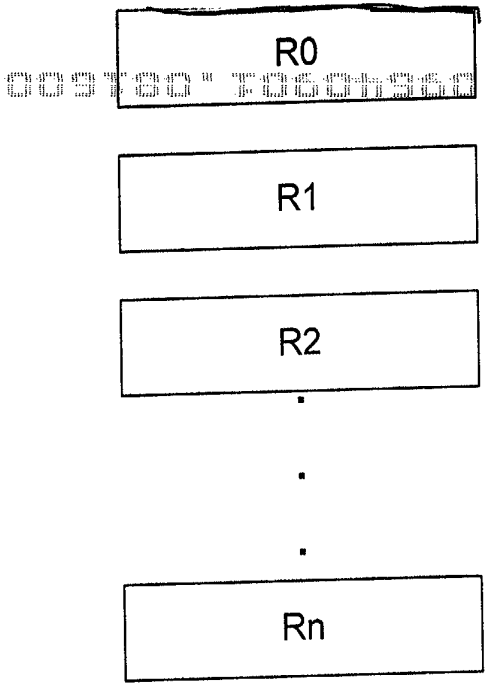


Fig. 2A

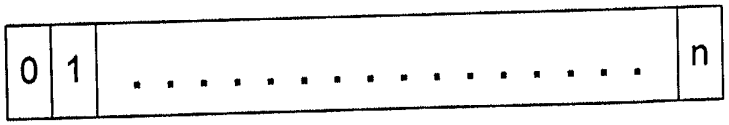


Fig. 2B

OPCODE	RD	RS1	RS2	RS3

Fig. 3A

OPCODE	RD	RS1	RS2
OPCODE	RD	RS1	immediate

Fig. 3B

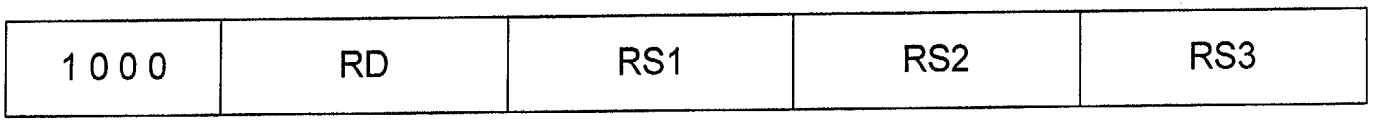


Fig. 4A

Fig. 4B

Fig. 4C

00100101010	RD	RS1	RS2
00101101010	RD	RS1	immediate

Fig. 4D

00100101001	RD	RS1	RS2
00101101001	RD	RS1	immediate

Fig. 4E

00100101000	RD	RS1	RS2
00101101000	RD	RS1	immediate

Fig. 4F

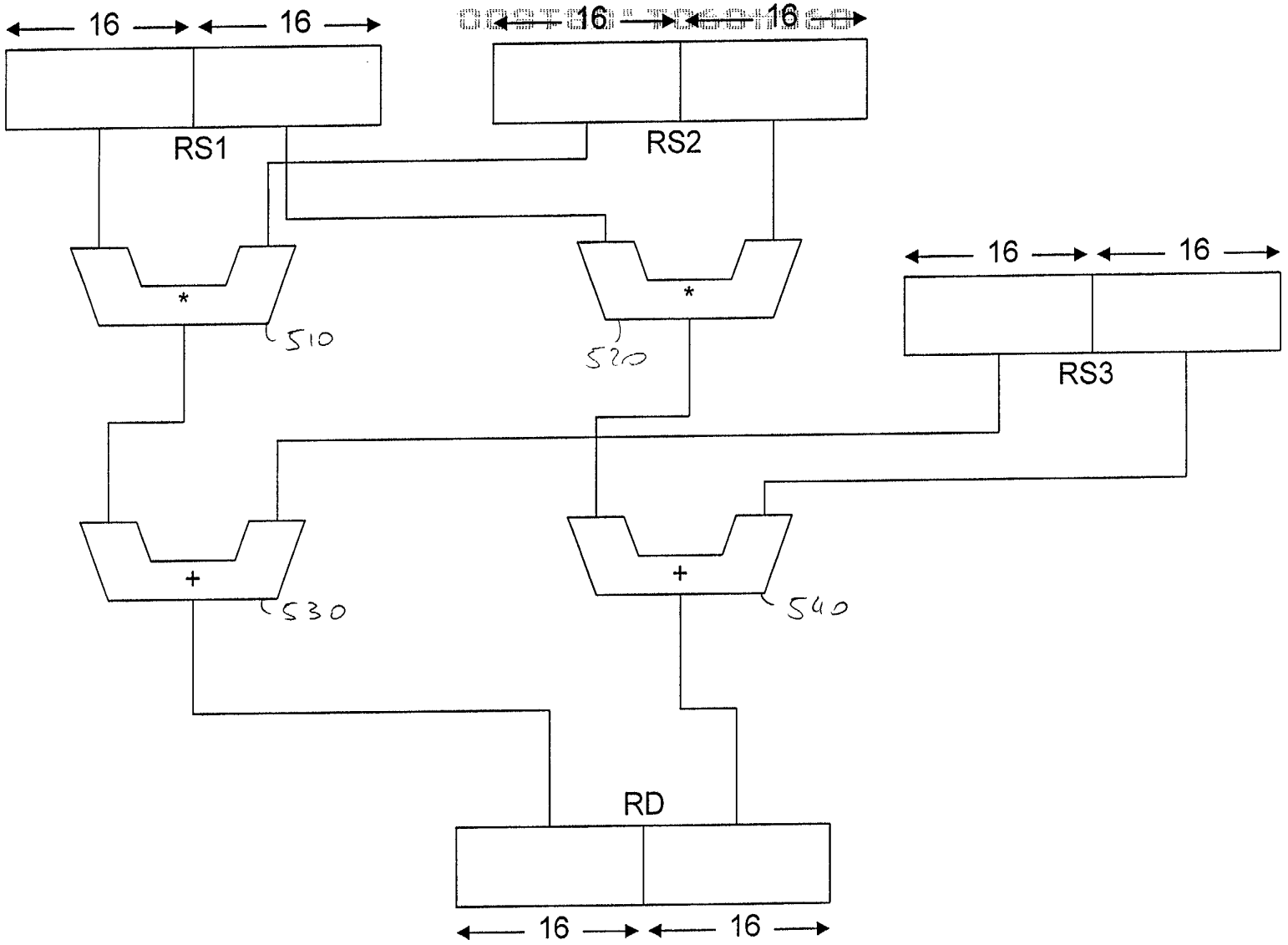


Fig. 5

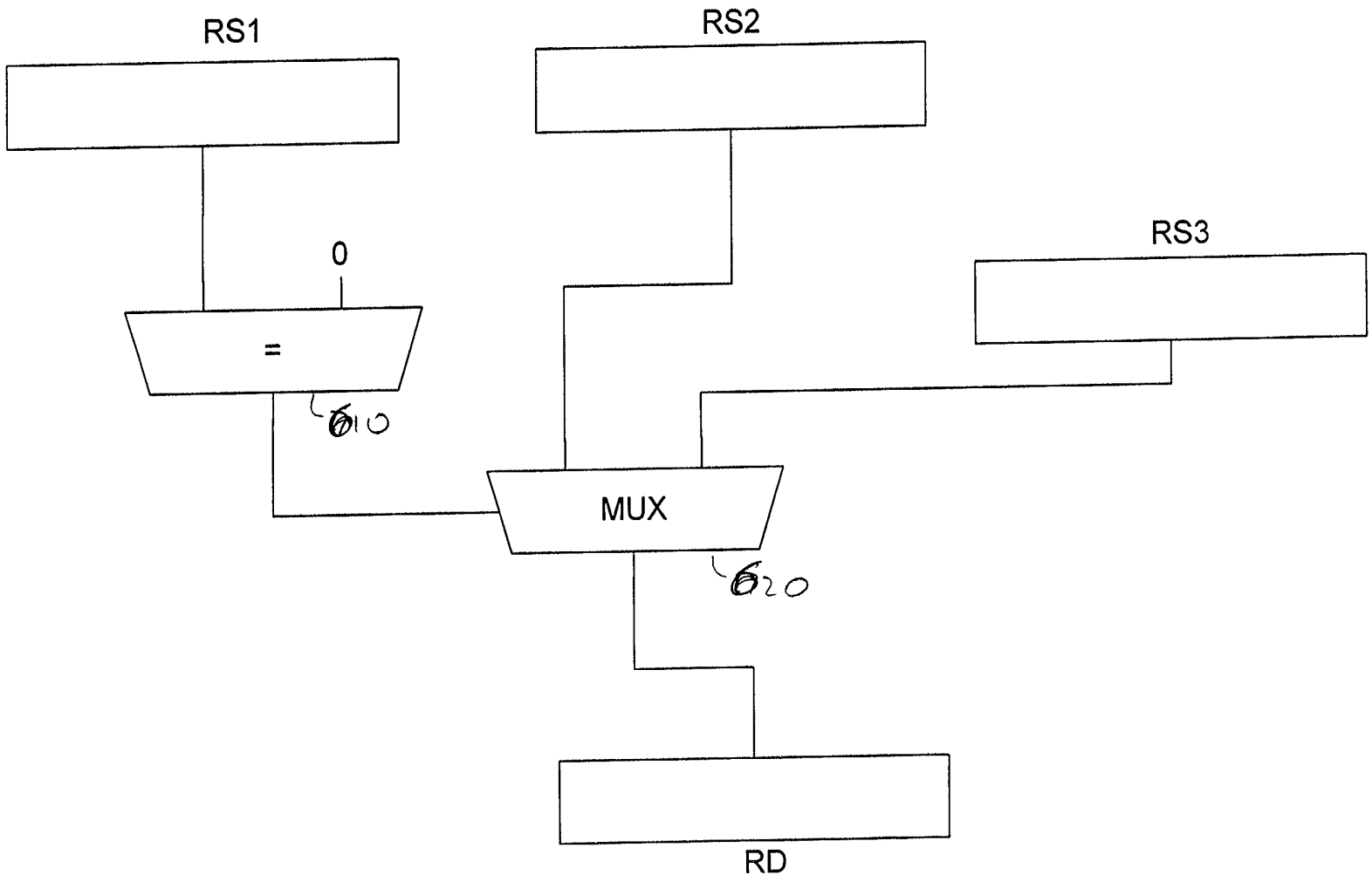


Fig. 6

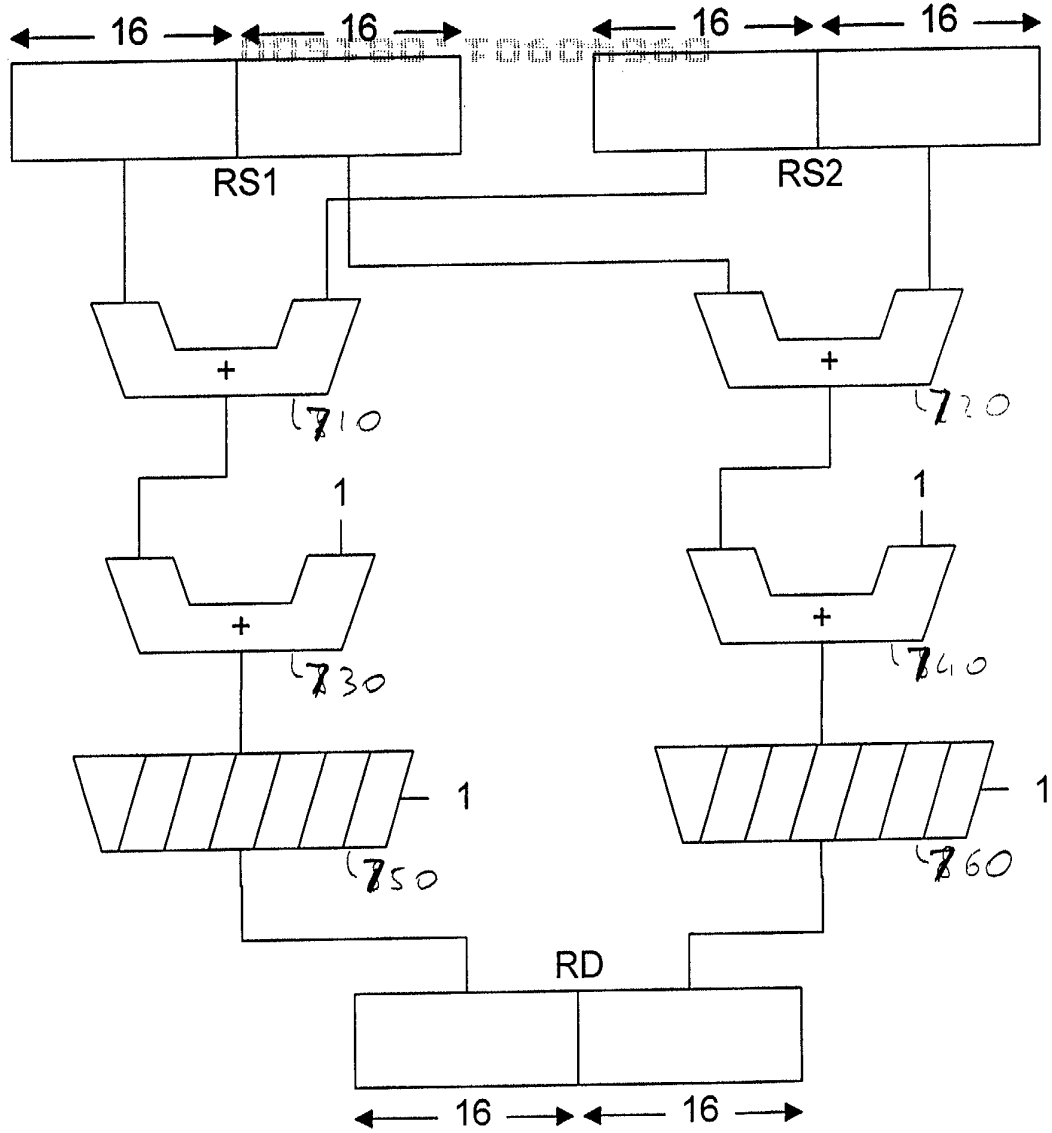


Fig. 7

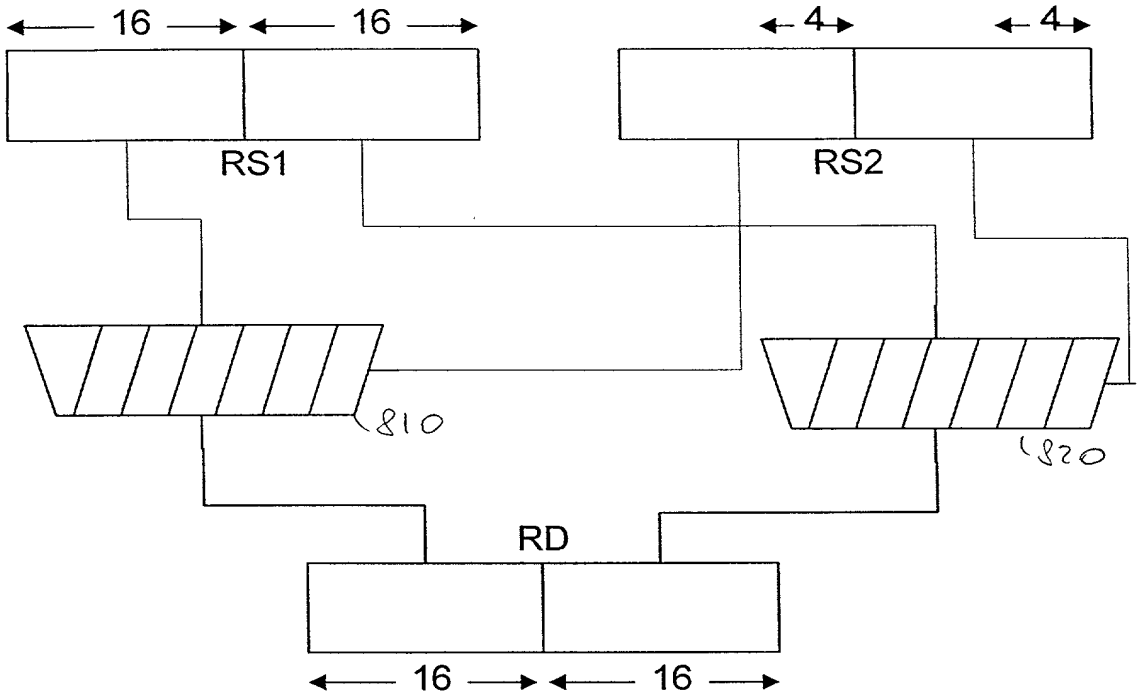

16
16
4
4
RS1
RS2
810
820
RD
16
16

Fig.8A

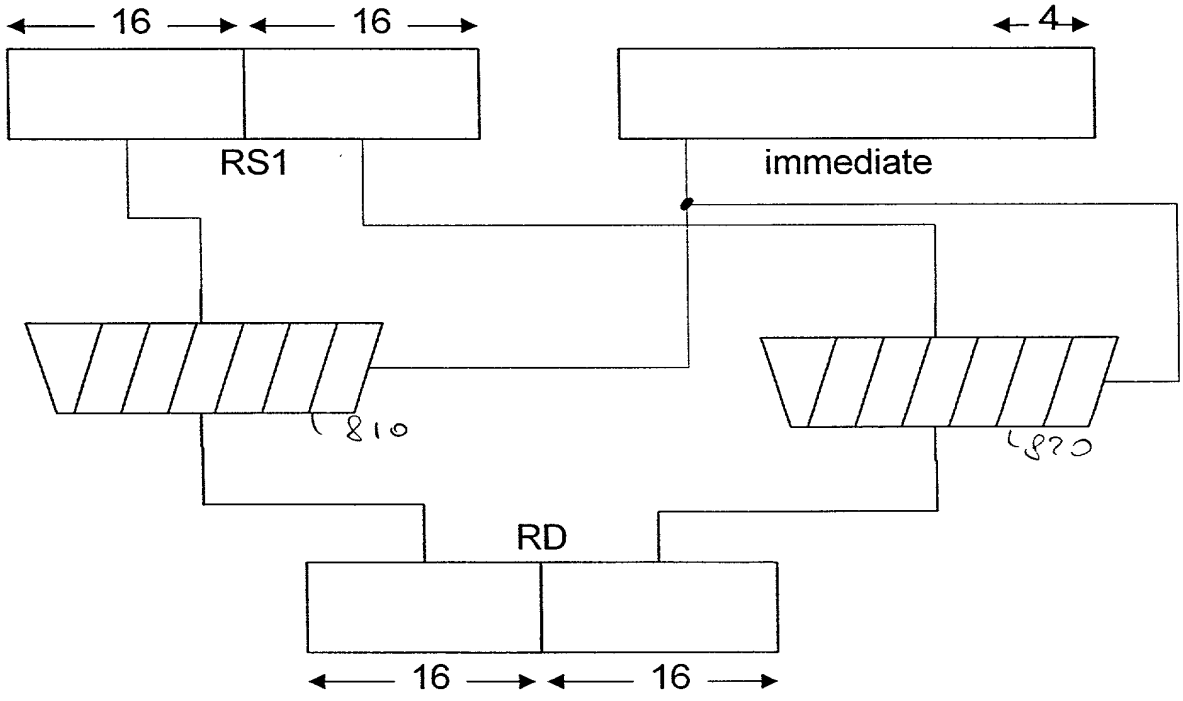

16
16
4
RS1
immediate
810
820
RD
16
16

Fig.8B

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original and joint inventor of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) which is claimed and for which a patent is sought by way of the application entitled

Accelerated Graphic Operations

which (check) ☒ is attached hereto.
☐ and is amended by the Preliminary Amendment attached hereto.
☐ was filed on _____ as Application Serial No. _____.
☐ and was amended on ___ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(d) of any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
Number	Country	Day/Month/Year Filed	Yes	No
N/A	N/A	N/A	☐	☒

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Provisional Application Number	Filing Date
N/A	N/A

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the

duty to disclose information, which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56, which became available between the filing date of the prior application(s) and the national or PCT international filing date of this application:

Application Serial No.	Filing Date	Status (patented, pending, abandoned)
N/A	N/A	N/A

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected therewith:

Alan H. MacPherson (24,423); Brian D. Ogonowsky (31,988); David W. Heid (25,875); Norman R. Klivans (33,003); Edward C. Kwok (33,938); David E. Steuber (25,557); Michael Shenker (34,250); Stephen A. Terrile (32,946); Peter H. Kang (40,350); Ronald J. Meetin (29,089); Ken John Koestner (33,004); Omkar K. Suryadevara (36,320); David T. Millers (37,396); Michael P. Adams (34,763); Robert B. Morrill (43,817); James E. Parsons (34,691); Philip W. Woo (39,880); Emily Haliday (38,903); Tom Hunter (38,498); Michael J. Halbert (40,633); Gary J. Edwards (41,008); Daniel P. Stewart (41,332); John T. Winburn (26,822); Tom Chen (42,406); Fabio E. Marino (43,339); Don C. Lawrence (31,975); Marc R. Ascolese (42,268); Carmen C. Cook (42,433); David G. Dolezal (41,711); Roberta P. Saxon (43,087); Mary Jo Bertani (42,321); Dale R. Cook (42,434); Sam G. Campbell (42,381); Matthew J. Brigham (44,047); Hugh H. Matsubayashi (43,779); Patrick D. Benedicto (40,909); T.J. Singh (39,535); Shireen Irani Bacon (40,494); Rory G. Bens (44,028); George Wolken, Jr. (30,441); John A. Odozynski (28,769); Cameron K. Kerrigan (44,826); Paul E. Lewkowicz (44,870); Theodore P. Lopez (44,881); Mayankkumar M. Dixit (44,064); Eric Stephenson (38,321); Christopher Allenby (45,906); David C. Hsia (46,235); Mark J. Rozman (42,117); Margaret M. Kelton (44,182); Do Te Kim (46,231); Alex Chen (45,591); Monique M. Heyninck (44,763); Gregory J. Michelson (44,940); Jonathan Geld (44,702); Emmanuel Rivera (45,760); Jason FarHadian (42,523); Matthew J. Spark (43,453); Elaine H. Lo (41,158) and Kenneth Olsen (26,493); Timothy J. Crean (37,116); Philip J. McKay (38,966); Robert S. Hauser (37,847); Joseph T. FitzGerald (33,881); Alexander E. Silverman (37,940); Christine S. Lam (37,489); Anirma Rakshpal Gupta (38;275); and Sean Patrick Lewis (42,798).

Please address all correspondence and telephone calls to:

Fabio E. Marino
Attorney for Applicants
SKJERVEN, MORRILL, MacPHERSON, FRANKLIN & FRIEL LLP
25 Metro Drive, Suite 700
San Jose, California 95110-1349

Telephone: 408-453-9200
Facsimile: 408-453-7979

I declare that all statements made herein of my own knowledge are true, all statements made herein on information and belief are believed to be true, and all statements made herein are made with the knowledge that whoever, in any matter within the jurisdiction of the Patent and Trademark Office, knowingly and willfully falsifies, conceals, or covers up by any trick, scheme, or device a material fact, or makes any false, fictitious or fraudulent statements or representations, or makes or uses any false writing or document knowing the same to contain any false, fictitious or fraudulent statement or entry, shall be subject to the penalties including fine or imprisonment or both as set forth under 18 U.S.C. 1001, and that violations of this paragraph may jeopardize the validity of the application or this document, or the validity or enforceability of any patent, trademark registration, or certificate resulting therefrom.

Full name of first joint inventor: Subramania Sudharsanan

Inventor's Signature: _____ Date: _____

Residence: Union City, California

Post Office Address: 4340 Cambridge Way
Union City, California 94587

Citizenship: Sri Lanka

Full name of second joint inventor: Jeffrey Meng Wah Chan

Inventor's Signature: _____ Date: _____

Residence: Mountain View, California

Post Office Address: 1984 Latham Street, Apt. 10
Mountain View, California 94040

Citizenship: Malaysian

Full name of third joint inventor: Michael F. Deering

Inventor's Signature: _____ Date: _____

Residence: Los Altos, California

Post Office Address: 657 Cuesta Drive
Los Altos, California 94024

Citizenship: U.S.A.

Full name of fourth joint inventor: Marc Tremblay

Inventor's Signature: _____ Date: _____

Residence: Palo Alto, California

Post Office Address: 801 Waverley Street, Apt. #3
Palo Alto, California 94301

Citizenship: Canada

Full name of fifth joint inventor: Scott R. Nelson

Inventor's Signature: _____ Date: _____

Residence: Pleasanton, California

Post Office Address: 4429 Clovewood Lane
Pleasanton, California 94588

Citizenship: U.S.A.